Self-Attentive Sequential Recommendation

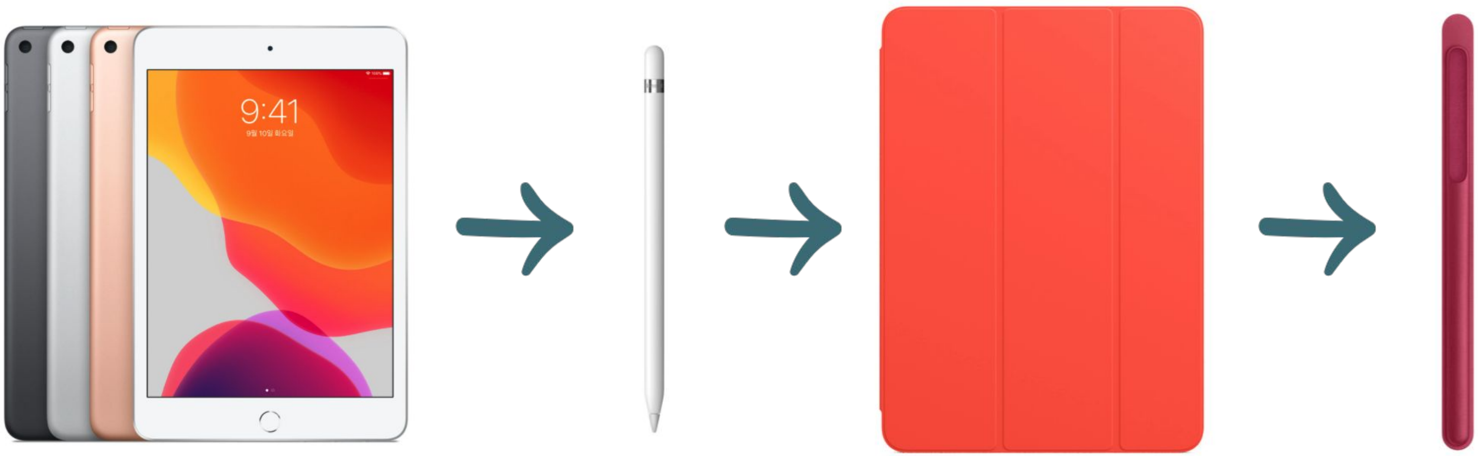
Wang-Cheng Kang, Julian McAuley (ICDM `18)

Hyunji Choi

June 7th, 2021

Sequential Recommendation

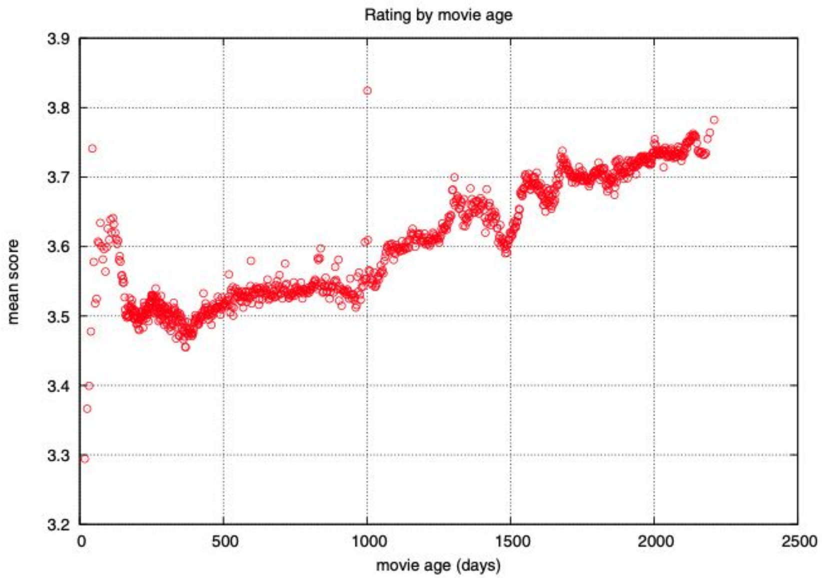
- **Combine personalized models of user behavior with context based on users' recent actions.**



www.apple.com

Sequential Recommendation

- **Time matters in Temporal Recommendation (ex. timeSVD++).**



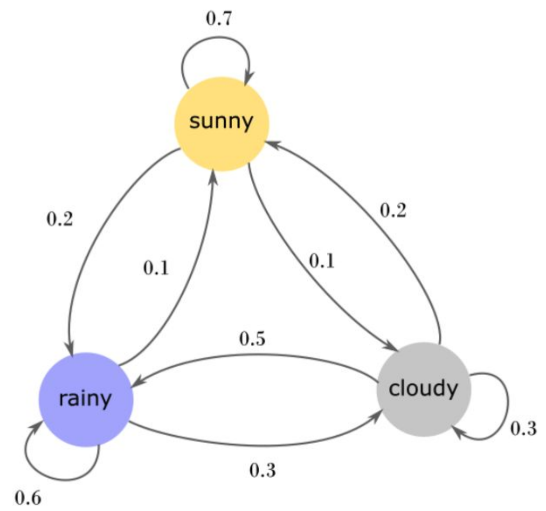
$$b_{ui}(t) = \mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} + (b_i + b_{i,\text{Bin}(t)}) \cdot c_u(t)$$

Collaborative Filtering with Temporal Dynamics

- **Order matters in Sequential Recommendation.**

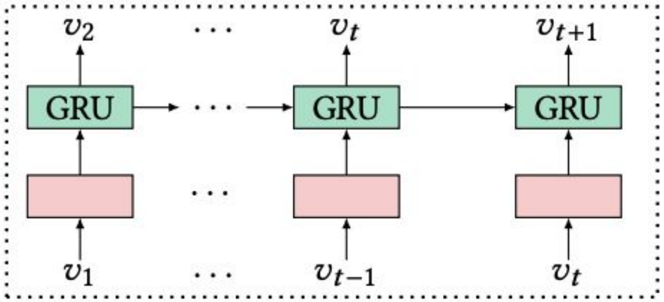
Two Approaches of Sequential Recommendation

- **Markov Chain**
  - Use last few activities.
  - Low order (<= 5) only.
  - Works well with sparse data.
  - P(sunny | rainy, cloudy, sunny, sunny)?



https://deparkes.co.uk/2020/08/08/markov-chains/

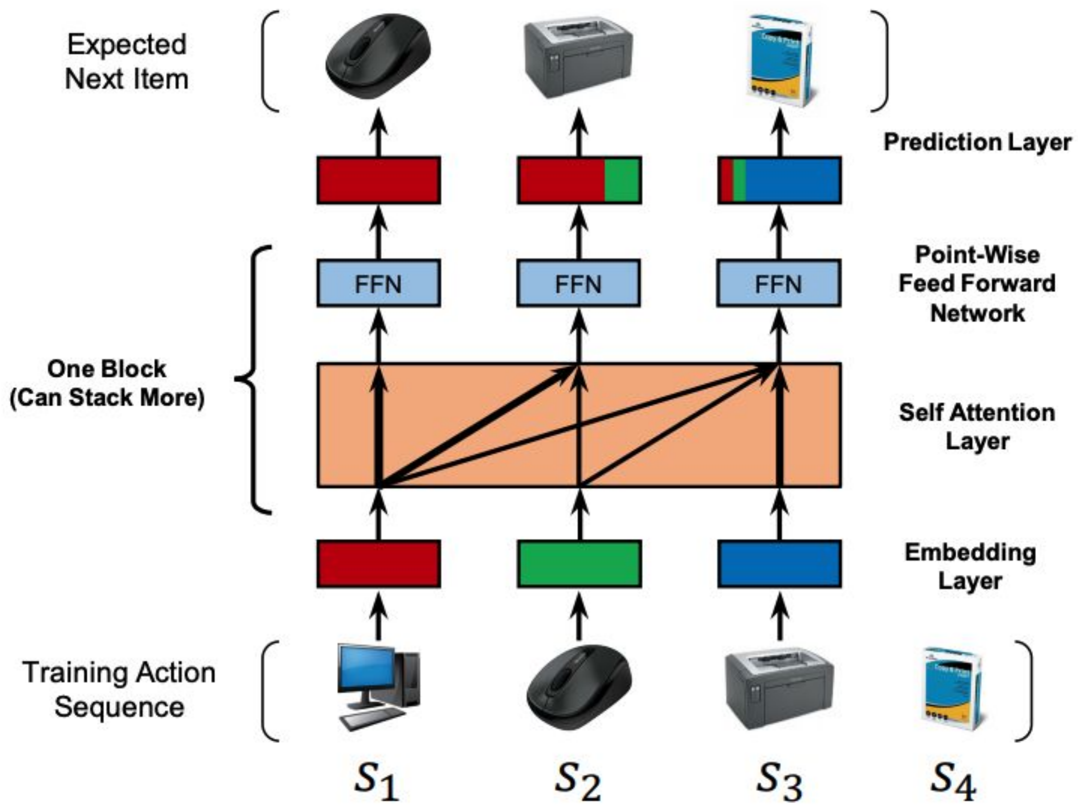
- **RNN**
  - Captures long-range context.
  - Works well in dense data.
  - Inefficient training time.



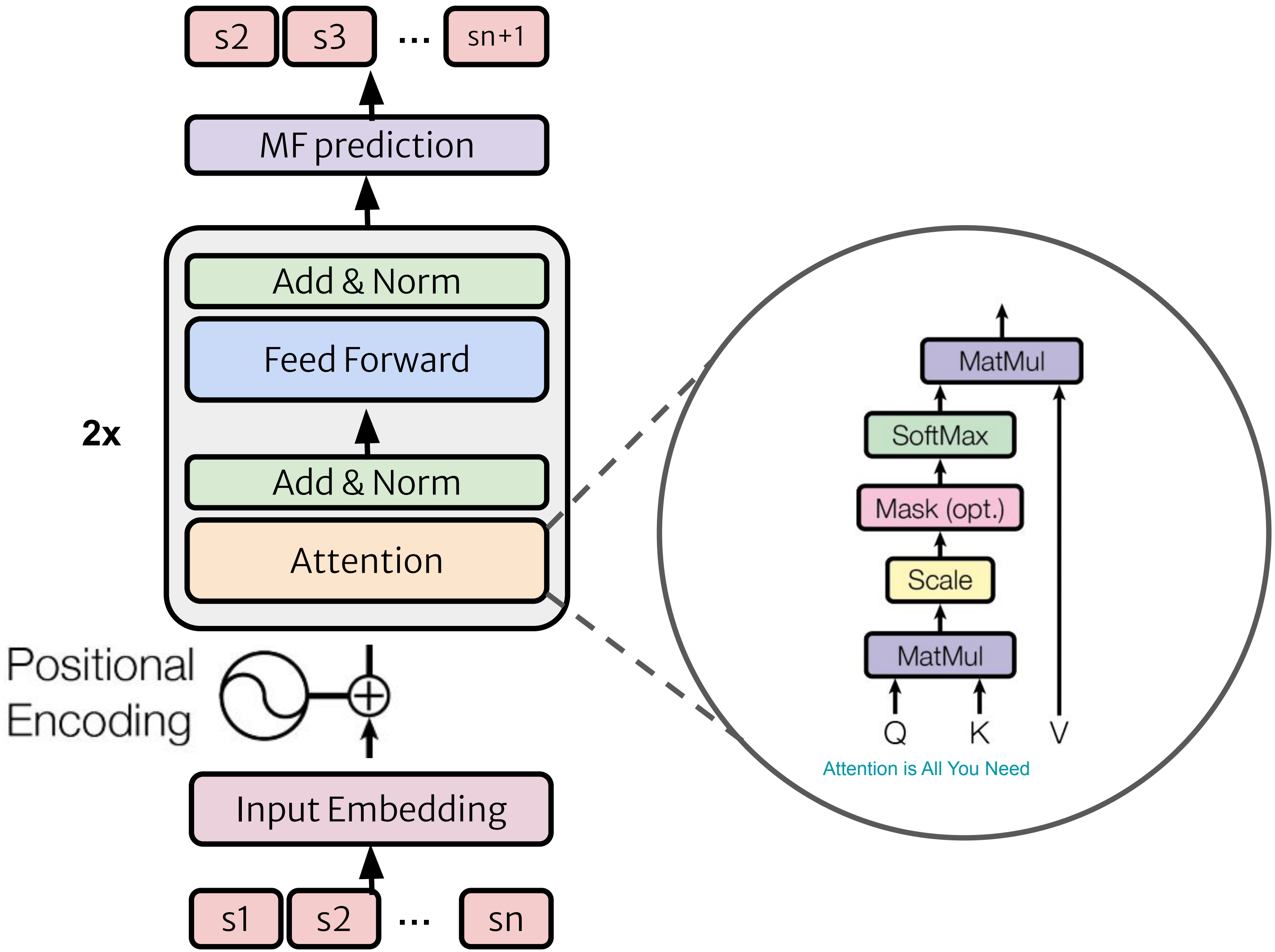
BERT4Rec

Self-Attentive Approach

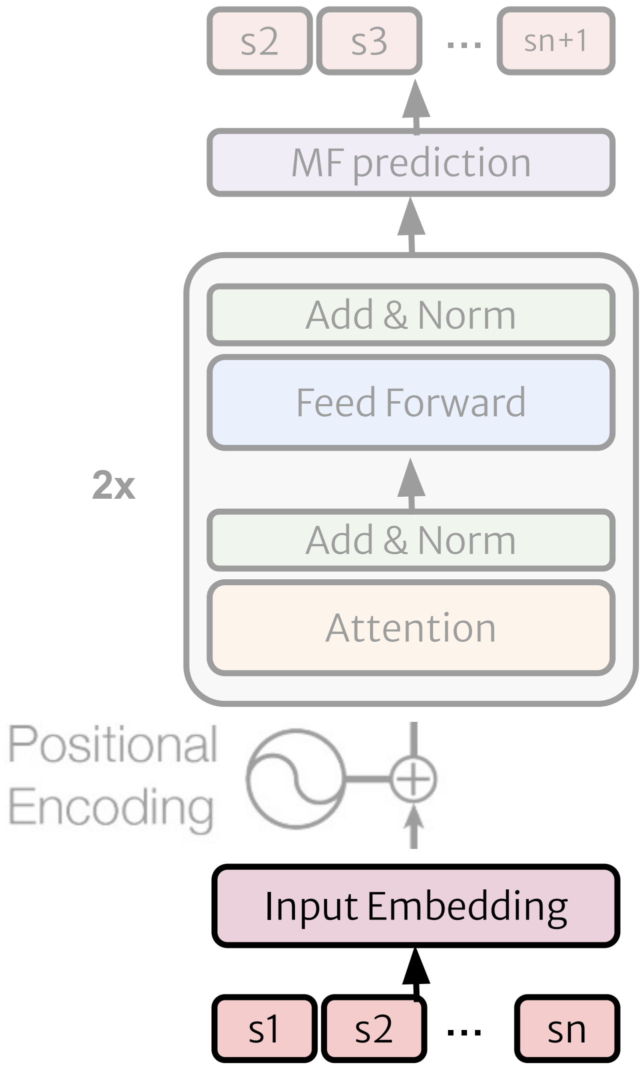
- **Capture long-term semantics + Select relatively few actions.**



Overview


s2
s3
...
sn+1
MF prediction
Add & Norm
Feed Forward
2x
Add & Norm
Attention
Positional Encoding
Input Embedding
s1
s2
...
sn
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Attention is All You Need

Embedding Layer



- **Sequence transformation**

  Maximum length of sequence: n = 5

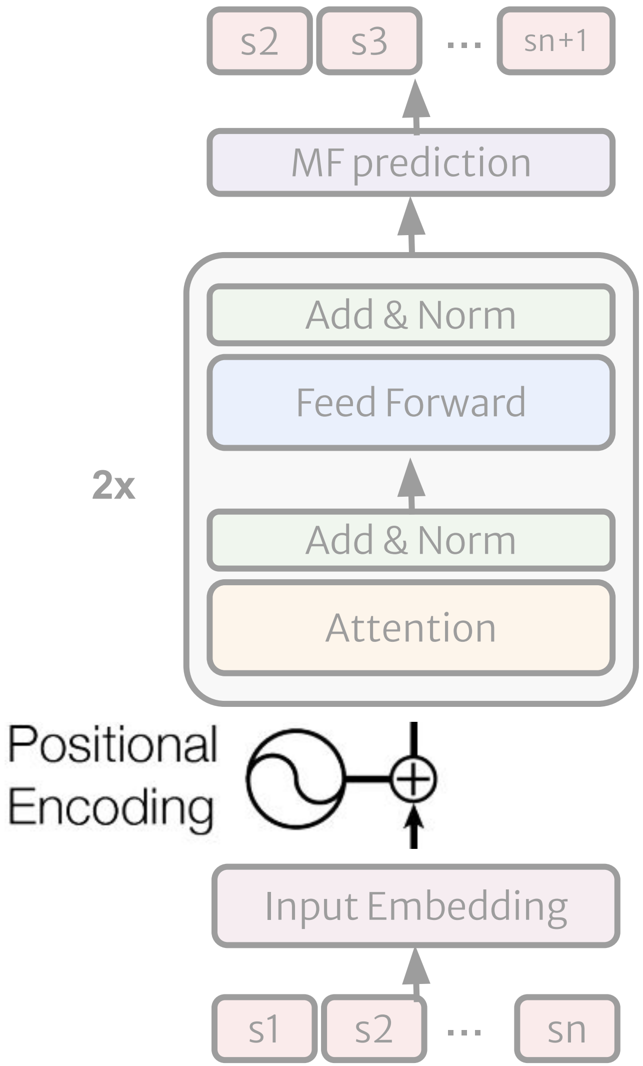
	0	0	a	b	c
a	b	c	d	e	f

- **Input embedding matrix**

  Item embedding dimension: d = 3

b	c	d	e	f
0.2	0.3	0.1	0.2	0.4
0.5	0.2	0.3	0.4	0.6
0.7	0.9	0.1	0.8	0.5

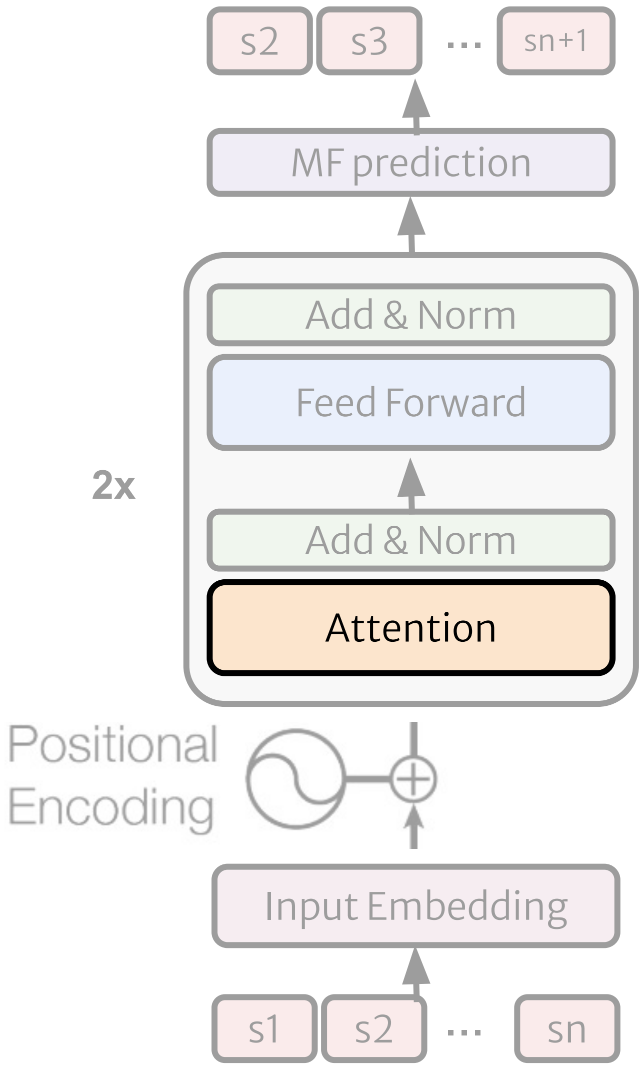
Embedding Layer



- **Positional embedding**

  Order info in n x d matrix: learnable

Self-Attention Layer



- **Scaled dot-product attention**

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

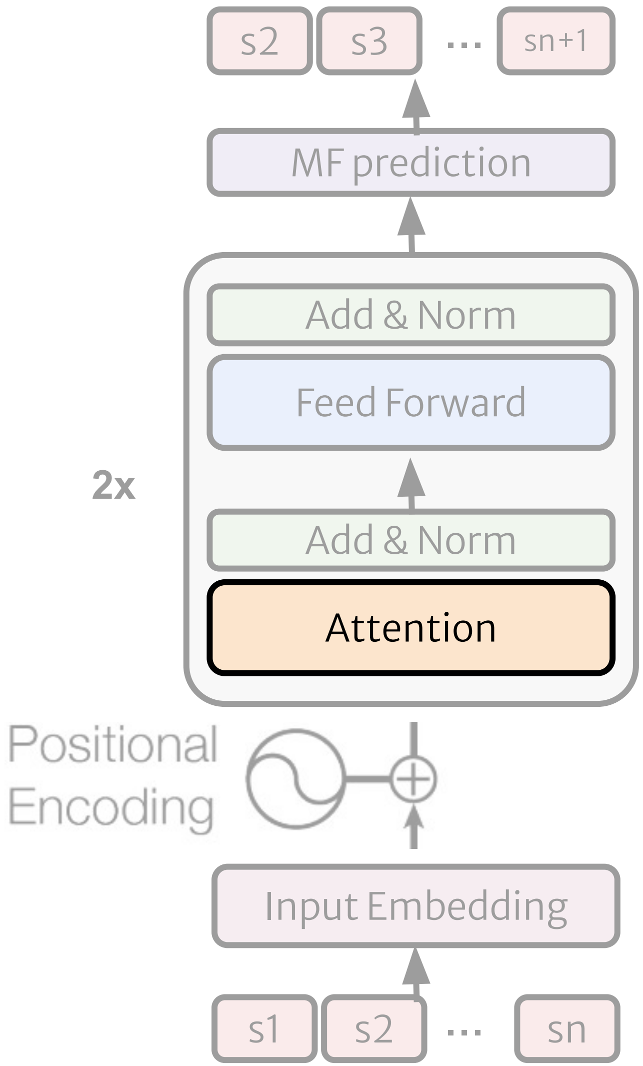
Weighted sum of V, where weight

is relevance of Q and K.

	k1	**k2**	**k3**	**k4**	**k5**
q1	0.2	0.3	0.1	0.2	0.4
q2	0.5	0.2	0.3	0.4	0.6
q3	0.7	0.9	0.1	0.8	0.5

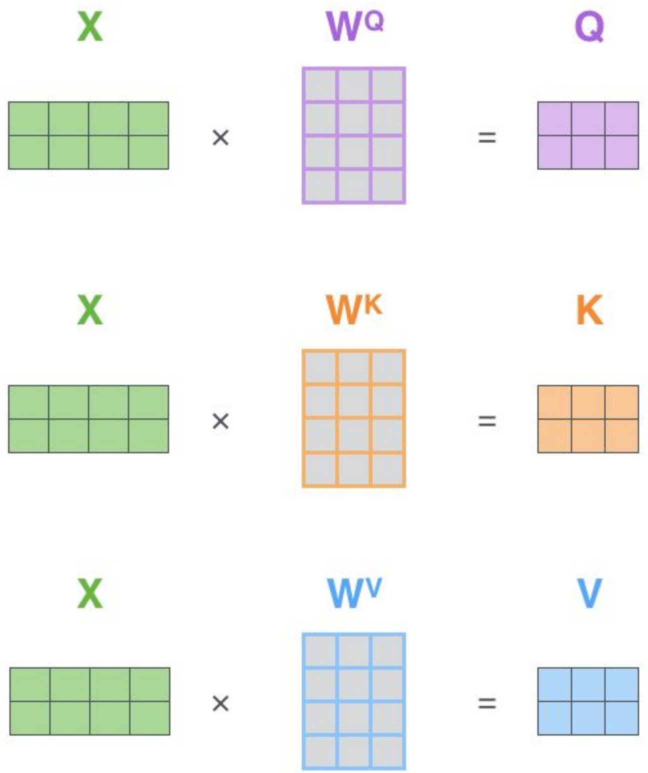
v1	0.2	0.3
v2	0.5	0.2
v3	0.7	0.9
v4	0.4	0.3
v5	0.1	0.2

Self-Attention Layer



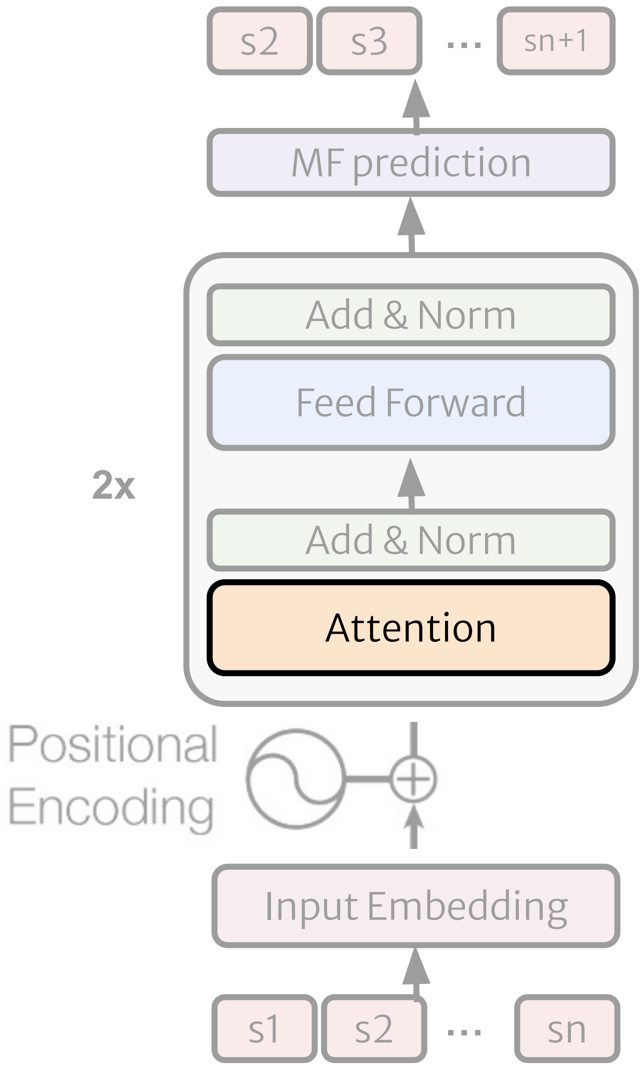
- **Linear projections**

$$\mathbf{S} = \mathrm{SA}(\widehat{\mathbf{E}}) = \mathrm{Attention}(\widehat{\mathbf{E}}\mathbf{W}^Q, \widehat{\mathbf{E}}\mathbf{W}^K, \widehat{\mathbf{E}}\mathbf{W}^V)$$



https://jalammar.github.io/illustrated-transformer/

Self-Attention Layer

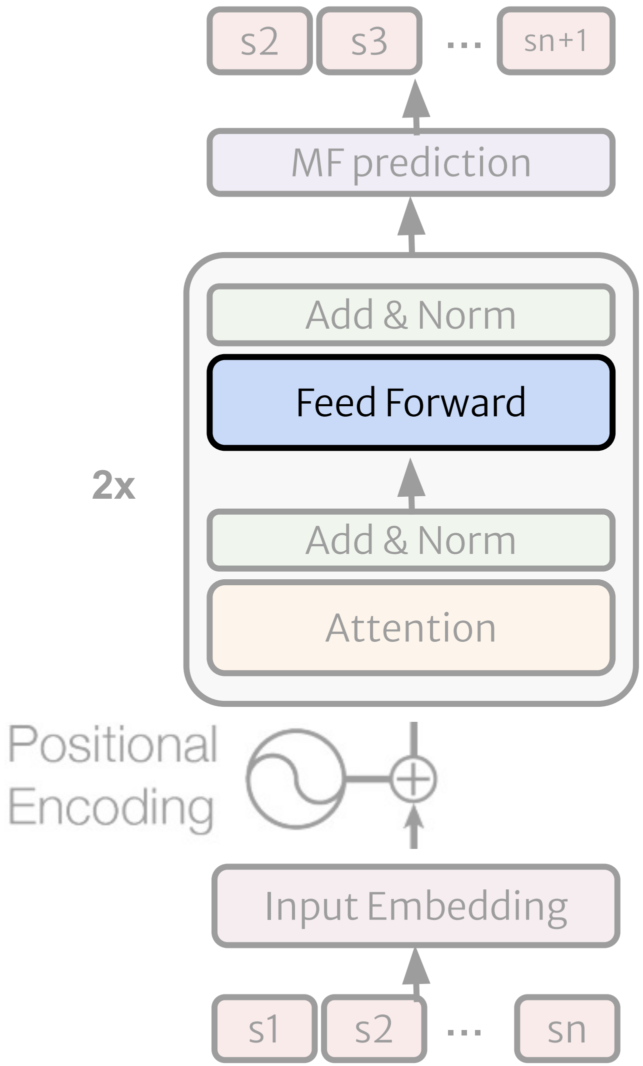


- **Causality masking**

	k1	k2	k3
q1	0.9	-99	-99
q2	0.5	0.9	-99
q3	0.7	0.7	1.0

v1	0.2	0.3
v2	0.5	0.2
v3	0.7	0.9

Point-Wise Feed-Forward Network

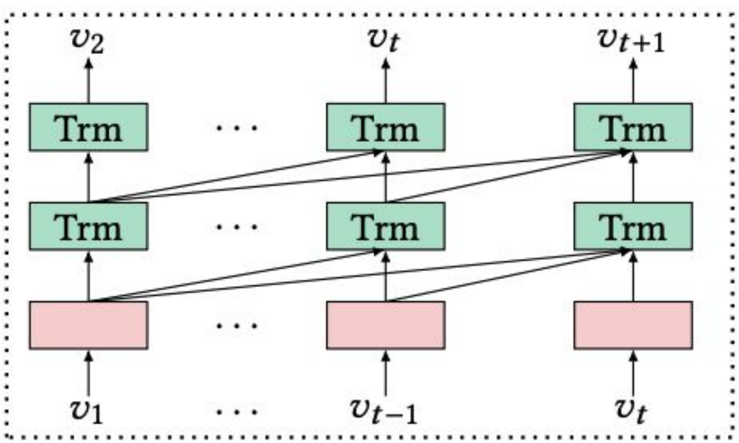


- **Non-linearity**
  - Si share weights.
  - Layers do not share weights.
  - Si and Sj have no interactions.

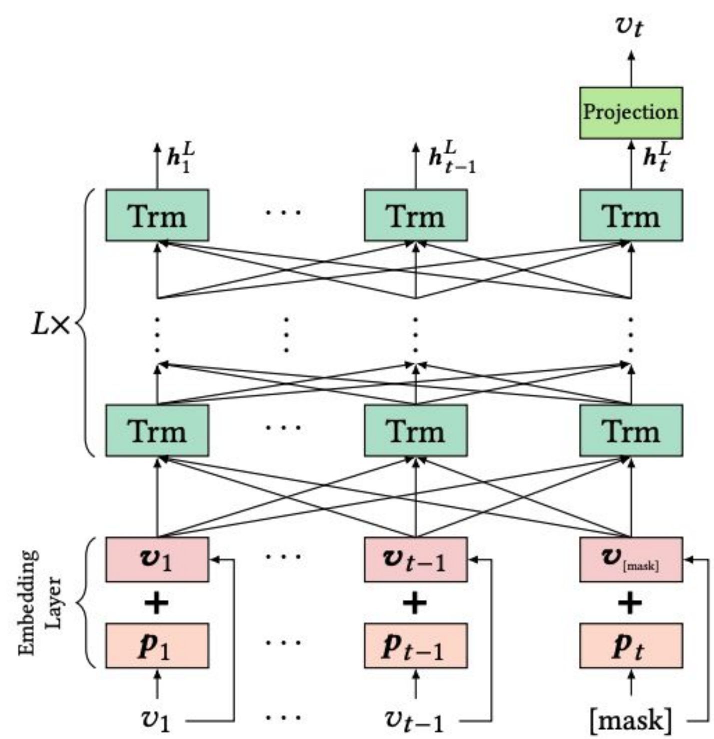
$$\mathbf{F}_i = \mathrm{FFN}(\mathbf{S}_i) = \mathrm{ReLU}(\mathbf{S}_i \mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

SASRec vs BERT4Rec

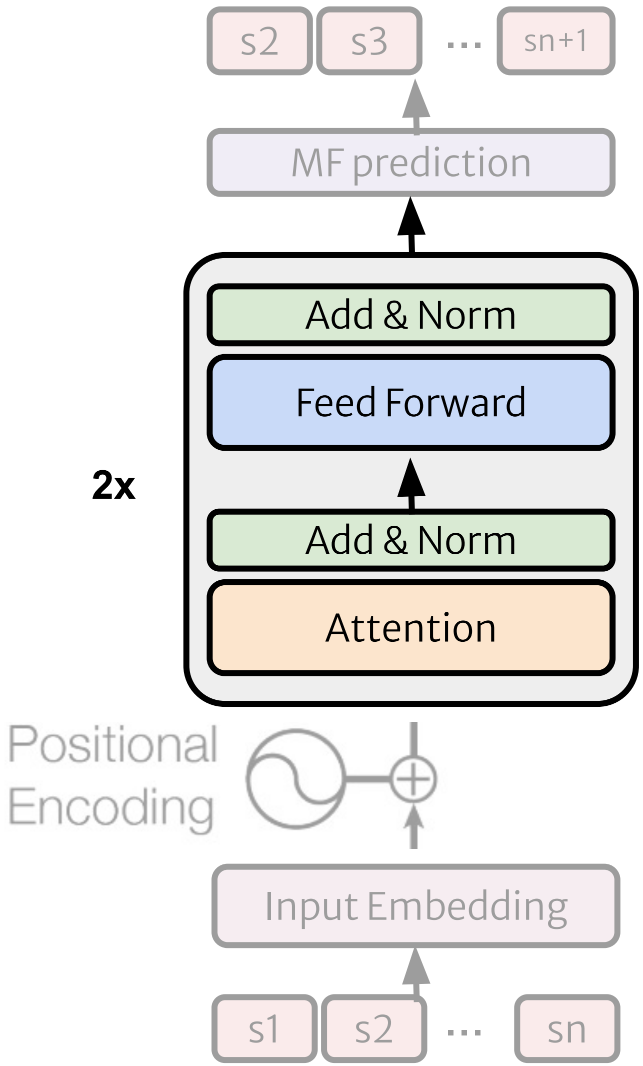
- **SASRec**
  - Uni-directional.
  - Causality masking.
- **BERT4Rec**
  - Bi-directional.
  - Cloze task.



BERT4Rec

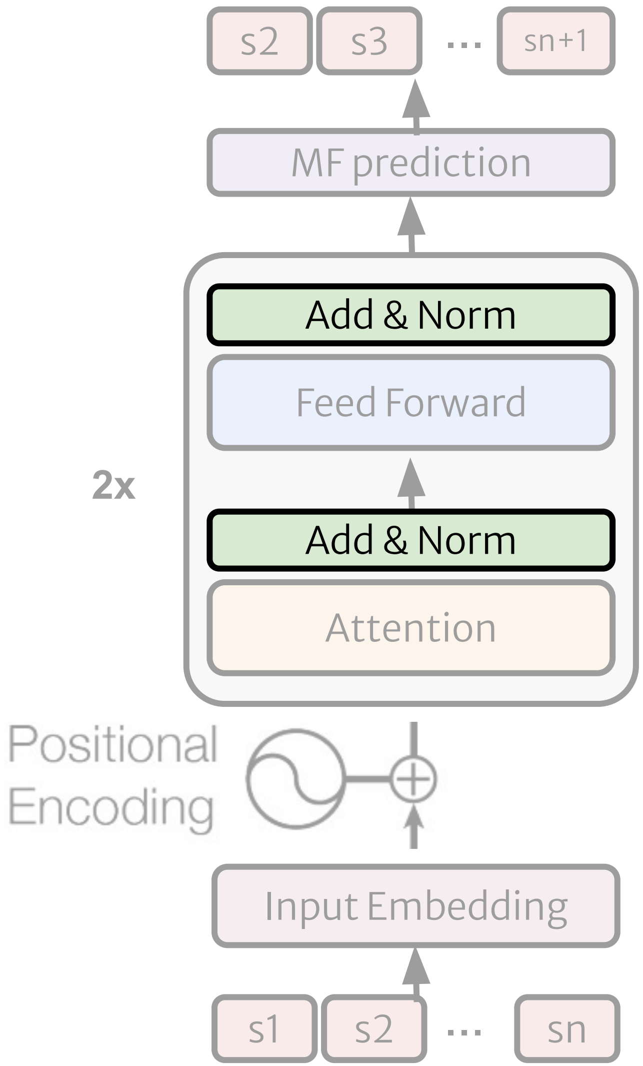


Stacking Self-Attention Blocks



- **Learn high-order item transactions.**
- **Problems**
  - Overfitting
  - Vanish gradients
  - More training time

Stacking Self-Attention Blocks



$$g(x) = x + \text{Dropout}(g(\text{LayerNorm}(x)))$$

- **Residual Connections**

  Propagate low-layer features.

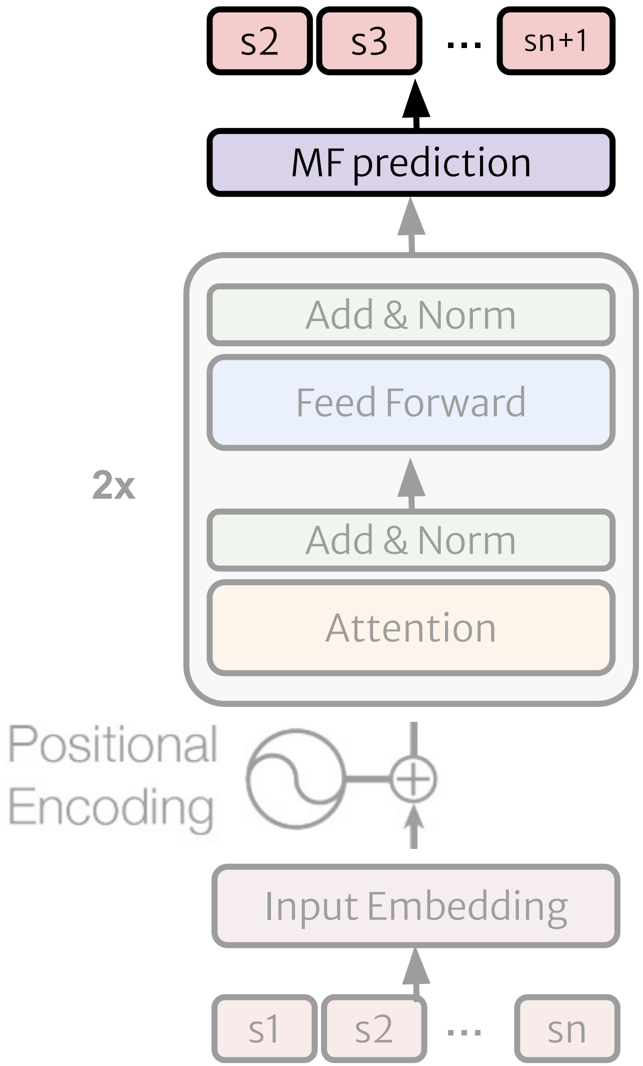
- **Layer Normalization**

  Stabilize and Accelerate.

- **Dropout**

  Prevent overfitting.

Prediction Layer



- **Matrix Factorization**

  Relevance of item i given first t items:

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{N}_i^T$$

- **Shared item embedding**

  Reduce model size, alleviate overfitting.

Training

- **Objective function: binary cross entropy loss**

$$-\sum_{\mathcal{S}^u \in \mathcal{S}} \sum_{t \in [1,2,\ldots,n]} \left[\log(\sigma(r_{o_t,t})) + \sum_{j \notin \mathcal{S}^u} \log(1 - \sigma(r_{j,t})) \right]$$

For all users and timestamp **Ground truth score** **Negative sample score**

- **Time complexity:** $O(n^2 d + n d^2)$
  - Fully parallelizable self-attention layer.
  - Ten times faster than CNN, RNN based models.
  - Easily scale n to a few hundred.

Data and Metric

- **Amazon Beaty, Games: high sparsity.**
- **Steam**
- **MovieLens-1M: dense.**

- **Hit Rate@10: GT in top 10.**
- **NDCG@10: larger weights on higher positions.**

Model Comparison

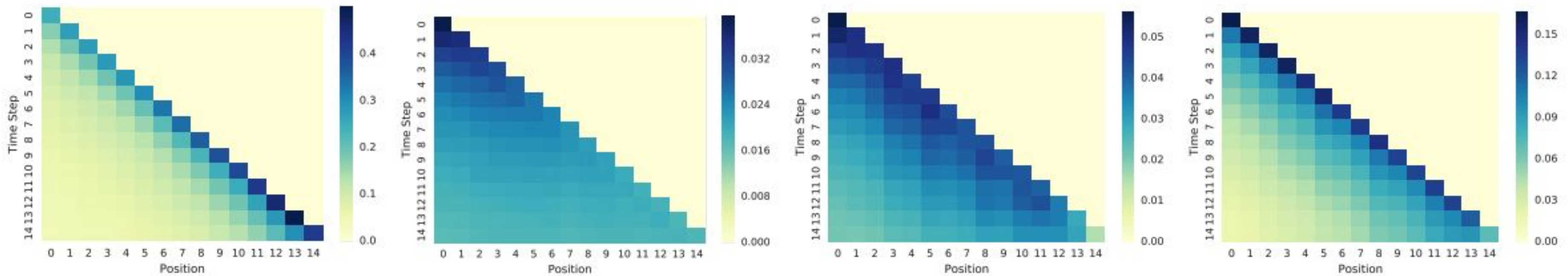
- **General**
  - PopRec
  - Bayesian Personalized Ranking
- **First order Markov chain**
  - Factorized Markov Chains
  - Factorized personalized Markov Chains
  - Translation-based Recommendation
- **RNN/CNN based**
  - GRU4Rec
  - GRU4Rec+
  - Convolutional Sequence Embeddings

SASRec shows SOTA recommendation performance

Dataset	Metric	(a) PopRec	(b) BPR	(c) FMC	(d) FPMC	(e) TransRec	(f) GRU4Rec	(g) GRU4Rec^{+}	(h) Caser	(i) SASRec	Improvement vs. (a)-(e)	Improvement vs. (f)-(h)
Beauty	Hit@10	0.4003	0.3775	0.3771	0.4310	0.4607	0.2125	0.3949	0.4264	**0.4854**	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	0.3020	0.1203	0.2556	0.2547	**0.3219**	6.6%	25.9%
Games	Hit@10	0.4724	0.4853	0.6358	0.6802	0.6838	0.2938	0.6599	0.5282	**0.7410**	8.5%	12.3%
	NDCG@10	0.2779	0.2875	0.4456	0.4680	0.4557	0.1837	0.4759	0.3214	**0.5360**	14.5%	12.6%
Steam	Hit@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	0.8018	0.7874	**0.8729**	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	0.5595	0.5381	**0.6306**	21.4%	12.7%
ML-1M	Hit@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	0.7886	**0.8245**	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	0.5538	**0.5905**	14.1%	6.6%

- **Better than all 8 models.**
- **Adaptively attend items within different ranges.**

Attention works on positions



(a) *Beauty*, Layer 1 (b) *ML-1M*, Layer 1, w/o PE (c) *ML-1M*, Layer 1 (d) *ML-1M*, Layer 2

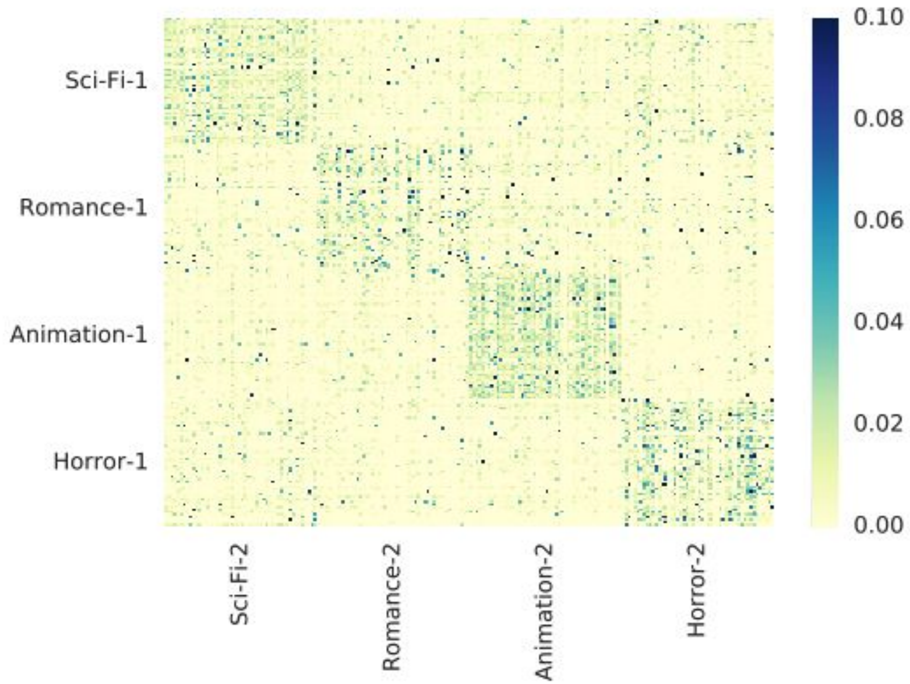
- **(a)-(c) Adaptive attention to dataset types.**
- **(b)-(c) Effect of positional embeddings.**
- **(c)-(d) Higher block attends to more recent items.**

Ablation Study

Architecture	*Beauty*	*Games*	*Steam*	*ML-1M*
(0) Default	0.3142	0.5360	0.6306	0.5905
(1) Remove PE	**0.3183**	0.5301	0.6036	0.5772
(2) Unshared IE	0.2437↓	0.4266↓	0.4472↓	0.4557↓
(3) Remove RC	0.2591↓	0.4303↓	0.5693	0.5535
(4) Remove Dropout	0.2436↓	0.4375↓	0.5959	0.5801
(5) 0 Block (b=0)	0.2620↓	0.4745↓	0.5588↓	0.4830↓
(6) 1 Block (b=1)	0.3066	**0.5408**	0.6202	0.5653
(7) 3 Blocks (b=3)	0.3078	0.5312	0.6275	**0.5931**
(8) Multi-Head	0.3080	0.5311	0.6272	0.5885

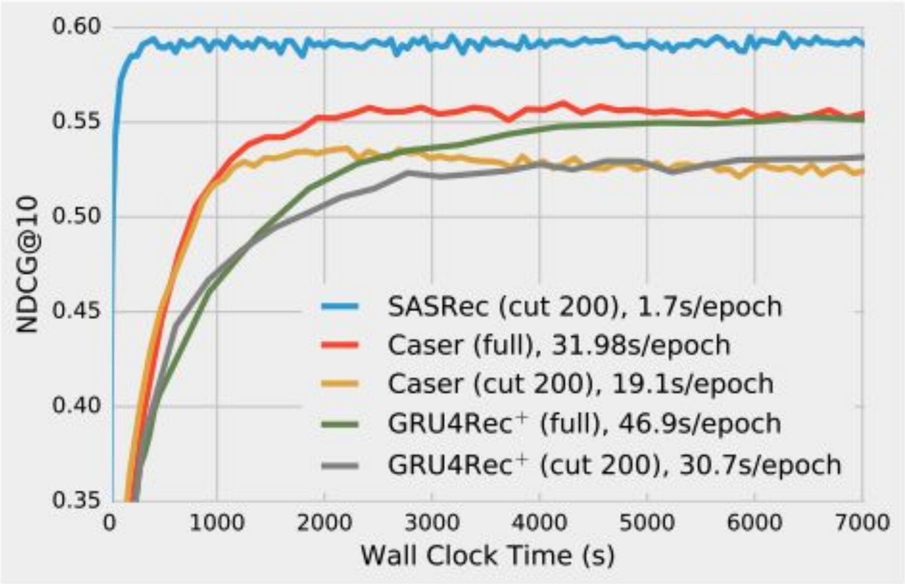
- **Positional embedding is important in dense dataset.**
- **Last few features are critical in sparse dataset.**
- **Dropout, sharing item embedding prevents overfitting.**

Attention works on items



- **Attention mechanism can identify similar items.**

SASRec is efficient and scalable



n	10	50	100	200	300	400	500	600
Time(s)	75	101	157	341	613	965	1406	1895
NDCG@10	0.480	0.557	0.571	0.587	0.593	0.594	0.596	0.595

- **SASRec runs and converges fast.**
- **Easily scale to a few hundred actions.**

Conclusion

- A novel self-attention based sequential model.
- Models the entire user sequence and with adaptive, position-aware, and hierarchical item similarity model.
- An order of magnitude faster than CNN/RNN based approaches due to fully parallelizable attention layer.